LAPORAN PRAKTIKUM DATA STRUCTURE

STACKS AND QUEUES

Dosen Pengampu:

H. Fatchurrochman,M.Kom

Asisten Praktikum:

Fillah Anjany 230605110033

Oleh :

Muhammad Alif Mujaddid **240605110082**

Jurusan Teknik Informatika Fakultas Sains dan Teknologi

UIN Maulana Malik Ibrahim Malang

2025

A. Praktikum

1. Implementasi Stack

   Salah satu contoh program sederhana yang mengimplementasikan stack adalah program pembalik kata. Stack digunakan untuk membalik huruf. Langkah pertama, tiap katakter pada String input diekstrak dan dimasukkan kedalam stack. Kemudian tiap karakter tersebut dikeluarkan dan ditampilkan sebagai output. Karena memiliki sifat LIFO, maka keluaran stack adalah karakter-karakter dengan urutan yang berkebalikan dengan input. Buatlah program pembalik kata tersebut dengan membuat 3 class, yaitu:

   a. class “stack”. Class ini digunakan untuk menyimpan setiap karakter input pada stack array. Berisi constructor dan method-method operasi stack.

   b. class “pembalik”. Class ini digunakan untuk membaca setiap karakter input, menyimpan karakter dengan memanggil method push() pada class “stack” (point a), dan membalik input dengan memanggil method pop() pada class “stack” (point a). Tiap karakter keluaran dari stack tersebut disimpan/ditambahkan (append) pada String output sebagai keluaran yang akan ditampilkan.

   c. class “AppPembalik”. Class ini berisi method main. Digunakan untuk deklarasi dan inisialisasi input, memanggil class “pembalik” untuk membalik input dan mendapatkan output kata yang telah dibalik, serta menampilkan output pada console. Untuk membaca tiap karakter String, anda dapat mengunakan method charAt() yang terdapat pada class String, misal StringInput.charAt(index).

   Contoh program pembalik kata ditunjukkan pada Gambar 3.1. Anda dapat mengerjakan sebagaimana Gambar 3.1 (a) dan point tambahan akan diberikan jika Anda mengerjakan sebagaimana Gambar 3.1 (b).

```
run:
>> katanya...
        kasur
>> dibalik jadi...
        rusak
BUILD SUCCESSFUL (total time: 2 seconds)
```

(a)

```
run:
Masukkan kata: bakso
kebalikan: oskab
Masukkan kata: soto
kebalikan: otos
Masukkan kata:
```

(b)

Gambar 3.4 Contoh output program pembalik kata. (a) String input diinisialisasi secara langsung pada listing program (hardcode). (b) String input didapat dari masukan dengan keyboard, program dapat membaca

dan membalikkan input secara berulang-ulang.

Jawaban :

```java
class Stack2 {
    private int maxSize;
    private char[] stackArray;
    private int top;

    public Stack2(int size) {
        maxSize = size;
        stackArray = new char[maxSize];
        top = -1;
    }

    public void push(char item) {
        stackArray[++top] = item;
    }

    public char pop() {
        return stackArray[top--];
    }

    public boolean isEmpty() {
        return (top == -1);
    }
}
```

```java
class Pembalik {
    private String input;

    public Pembalik(String input) {
        this.input = input;
    }

    public String balik() {
        int size = input.length();
        Stack2 stack = new Stack2(size);
        for (int i = 0; i < size; i++) {
            stack.push(input.charAt(i));
        }
        String output = "";
        while (!stack.isEmpty()) {
            output += stack.pop();
        }
        return output;
    }
}
```

```java
public class AppPembalik {
    Run | Debug
    public static void main(String[] args) {
        String kata = "kasur";
        Pembalik pembalik = new Pembalik(kata);

        String hasil = pembalik.balik();

        System.out.println("Kata asli   : " + kata);
        System.out.println("Kata dibalik: " + hasil);
        Scanner sc = new Scanner(System.in);
        for (int i = 0; i < 2; i++) {
            System.out.print(s:"Kata asli   : ");
            kata = sc.nextLine();
            pembalik = new Pembalik(kata);
            hasil = pembalik.balik();
            System.out.println("Kata dibalik: " + hasil);
        }
    }
}
```

Output :

```
addid@LAPTOP-F80MDN28 MINGW64 /a/Code/DSA (main)
$  cd a:\\Code\\DSA ; /usr/bin/env A:\\Java\\jdk-
6 -XX:+ShowCodeDetailsInExceptionMessages -cp C:\
7\\redhat.java\\jdt_ws\\DSA_2f521fc6\\bin Praktik
Kata asli   : kasur
Kata dibalik: rusak
Kata asli   : soto
Kata dibalik: otos
Kata asli   : bakso
Kata dibalik: oskab
```

2. Implementasi Queue

Implementasi queue banyak didunia nyata, sebagaimana antrian. Buatlah program simulasi antrian dengan mengimplementasikan Queue. Simulasi antrian menunjukkan (lihat Gambar 3.2):
- penambahan objek pada daftar antrian. Lakukan beberapa kali hingga antrian penuh. Ketika objek ditambahkan pada antrian yang penuh maka program akan menampilkan keterangan antrian penuh.
- Menampilkan isi antrian
- Satu persatu objek keluar antrian hingga antrian kosong

```
run:
>> beberapa mulai mengantri
Andi masuk antrian
Ahmad masuk antrian
Satrio masuk antrian
Afrizal masuk antrian
Maaf sukran, antrian masih penuh
Maaf Mahmud, antrian masih penuh

>> isi antrian
Andi,Ahmad,Satrio,Afrizal,

>> satu persatu keluar antrian
Andi Keluar antrian
Ahmad,Satrio,Afrizal,Kosong,

Ahmad Keluar antrian
Satrio,Afrizal,Kosong,Kosong,

Satrio Keluar antrian
Afrizal,Kosong,Kosong,Kosong,

Afrizal Keluar antrian
Kosong,Kosong,Kosong,Kosong,

antrian kosong
0 Person
Kosong,Kosong,Kosong,Kosong,
BUILD SUCCESSFUL (total time: 2 seconds)
```

Gambar 3.5 Output program simulasi antrian

```
class Queue2 {
    private int maxSize;
    private String[] queArray;
    private int front;
    private int rear;
    private int nItems;

    public Queue2(int size) {
        maxSize = size;
        queArray = new String[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
    }

    public void insert(String value) {
        if (isFull()) {
            System.out.println("Maaf " + value + ", antrian masih penuh");
        } else {
            if (rear == maxSize - 1) {
                rear = -1;
            }
            queArray[++rear] = value;
            nItems++;
            System.out.println(value + " masuk antrian");
        }
    }
```

```
    public String remove() {
        if (isEmpty()) {
            return "Kosong";
        }
        String temp = queArray[front++];
        if (front == maxSize) {
            front = 0;
        }
        nItems--;
        System.out.println(temp + " Keluar antrian");
        return temp;
    }

    public boolean isEmpty() {
        return (nItems == 0);
    }

    public boolean isFull() {
        return (nItems == maxSize);
    }
```

```java
    public void tampil() {
        System.out.print(s:"Isi antrian: ");
        int count = 0;
        int i = front;
        while (count < nItems) {
            System.out.print(queArray[i] + ",");
            i++;
            if (i == maxSize) {
                i = 0;
            }
            count++;
        }
        // sisa kosong
        for (int j = nItems; j < maxSize; j++) {
            System.out.print(s:"Kosong,");
        }
        System.out.println();
    }
}
```

```java
    public static void main(String[] args) {
        Queue2 antrian = new Queue2(size:4);

        System.out.println(x:">> beberapa mulai mengantri");
        antrian.insert(value:"Andi");
        antrian.insert(value:"Ahmad");
        antrian.insert(value:"Satrio");
        antrian.insert(value:"Afrizal");
        antrian.insert(value:"Sukran"); // penuh
        antrian.insert(value:"Mahmud"); // penuh

        System.out.println(x:"\n>> isi antrian");
        antrian.tampil();

        System.out.println(x:"\n>> satu persatu keluar antrian");
        while (!antrian.isEmpty()) {
            antrian.remove();
            antrian.tampil();
        }

        System.out.println(x:"\nantrian kosong");
        antrian.tampil();
        System.out.println(x:"0 Person");
        antrian.tampil();
    }
}
```

Output :

```
addid@LAPTOP-F80MDN28 MINGW64 /a/Code/DSA (main)
$  cd a:\\Code\\DSA ; /usr/bin/env A:\\Java\\jdk-22\\bi
8 -XX:+ShowCodeDetailsInExceptionMessages -cp C:\\Users
7\\redhat.java\\jdt_ws\\DSA_2f521fc6\\bin Praktikum.Que
>> beberapa mulai mengantri
Andi masuk antrian
Ahmad masuk antrian
Satrio masuk antrian
Afrizal masuk antrian
Maaf Sukran, antrian masih penuh
Maaf Mahmud, antrian masih penuh

>> isi antrian
Isi antrian: Andi,Ahmad,Satrio,Afrizal,

>> satu persatu keluar antrian
Andi Keluar antrian
Isi antrian: Ahmad,Satrio,Afrizal,Kosong,
Ahmad Keluar antrian
Isi antrian: Satrio,Afrizal,Kosong,Kosong,
Satrio Keluar antrian
Isi antrian: Afrizal,Kosong,Kosong,Kosong,
Afrizal Keluar antrian
Isi antrian: Kosong,Kosong,Kosong,Kosong,

antrian kosong
Isi antrian: Kosong,Kosong,Kosong,Kosong,
0 Person
Isi antrian: Kosong,Kosong,Kosong,Kosong,
```

B. Kesimpulan

1. Konsep dan Implementasi Stack

Dari praktikum ini aku belajar kalau stack itu struktur data yang cara kerjanya mirip tumpukan barang. Prinsipnya LIFO (Last In First Out), artinya barang/data yang terakhir ditaruh di atas justru yang pertama kali diambil.

Operasi dasarnya ada push (nambahin data ke tumpukan), pop (ngambil data paling atas), dan peek (ngintip data paling atas tanpa ngambilnya).

Di praktikum, stack ini bisa dibuat pakai array. Contoh nyatanya, stack bisa dipakai buat membalik kata (misal "ABC" jadi "CBA") atau buat baca ekspresi matematika biar komputer ngerti urutan hitungannya.

2. Konsep dan Implementasi Queue

Kalau queue, cara kerjanya beda. Queue itu seperti antrian di kantin atau bioskop. Prinsipnya FIFO (First In First Out), jadi siapa yang datang duluan, dia yang dilayani lebih dulu.

Operasi dasarnya ada insert/enqueue (nambah data ke antrian), remove/dequeue (ngambil data paling depan), dan peek (lihat data paling depan).

Di praktikum, queue juga bisa dibuat pakai array. Dengan simulasi antrian, aku jadi lebih paham gimana data masuk ke belakang, keluar dari depan, sampai antriannya penuh atau kosong.